GOLANG </> TEMPL </> htmx

The GOTH Stack

Simple Web Applications

Workshop by Mirelle Bogdanski & Joel Eckhardt

ESDE 2024

Fontys Venlo

Pls clone the repo now and download the docker image!

Step 1: Scan the QR code

or

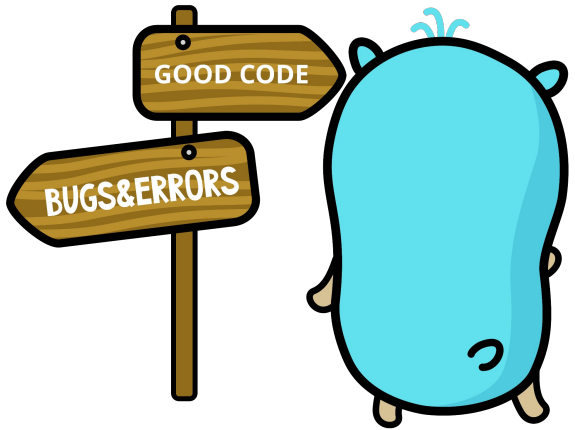
Type the URL into your browser

https://github.com/sebivenlo/esd-2024-goth-stack

Step 2: Open the root folder in VSCode, then reopen in container

Agenda

- Introduction into the GOTH stack (What and Why bother?)
- Intro into
  - Go(lang)
  - Templ
  - htmx
- (Go)ing further: Extending the GOTH stack
- Practical Part
- Quizz
- Wrap up

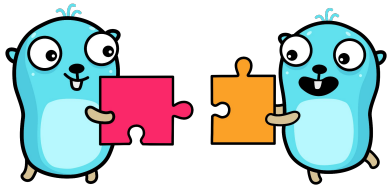


Introduction

What is the GOTH Stack?

- **Go** - for simple and performant web servers
- **Templ** - for HTML templating
- (sometimes Tailwind)
- **htmx** - for interactivity without requiring JS

→ Each component is about **simplicity and control**



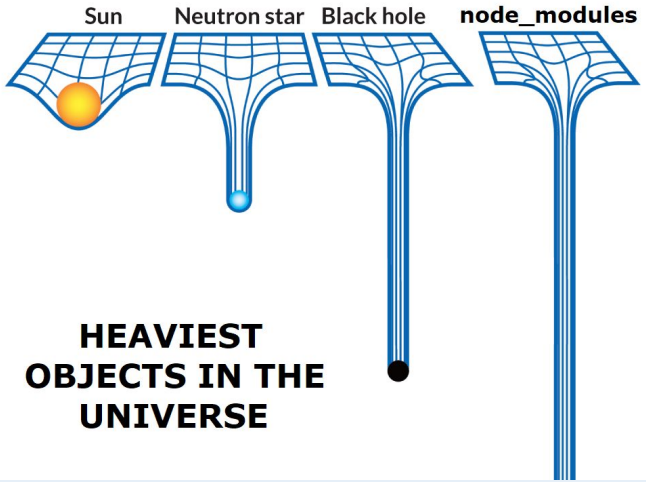
GOLANG
+
</> TEMPL
+
</> htmx

Introduction

Why use the GOTH Stack?

- Insert node_modules meme here
- **KISS approach** to web dev
- Server side rendering*
- State management purely on the server

→ Reduced complexity, fewer moving parts (no client side state management), high performance, control and security

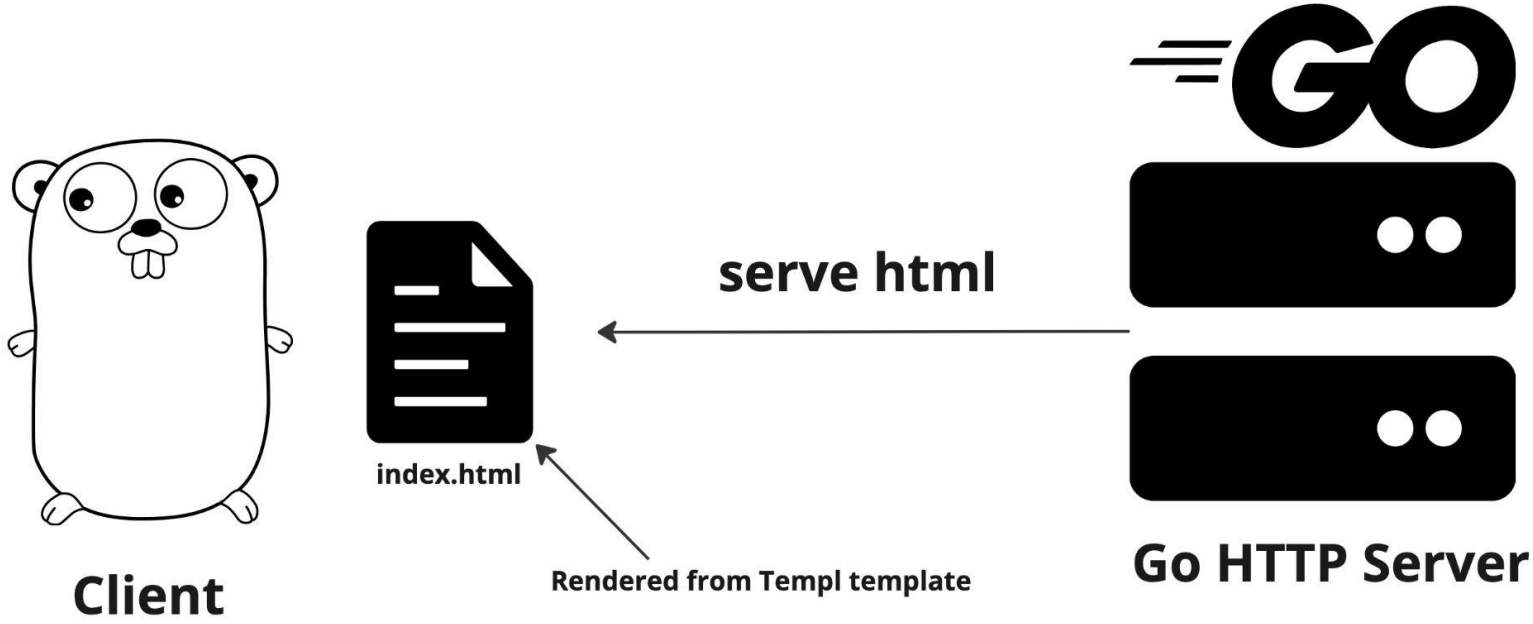


Introduction

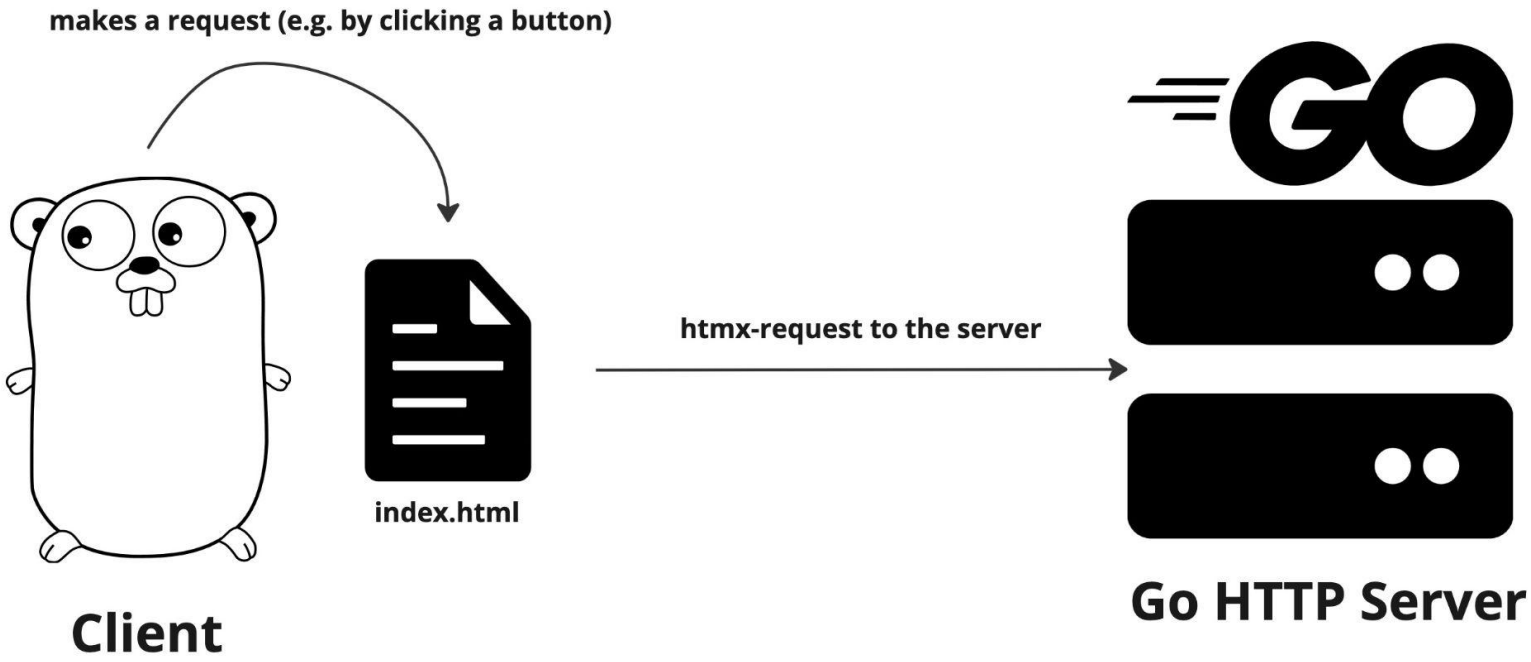
Okay, but why Go specifically?

We could achieve each of the benefits with another language, right?

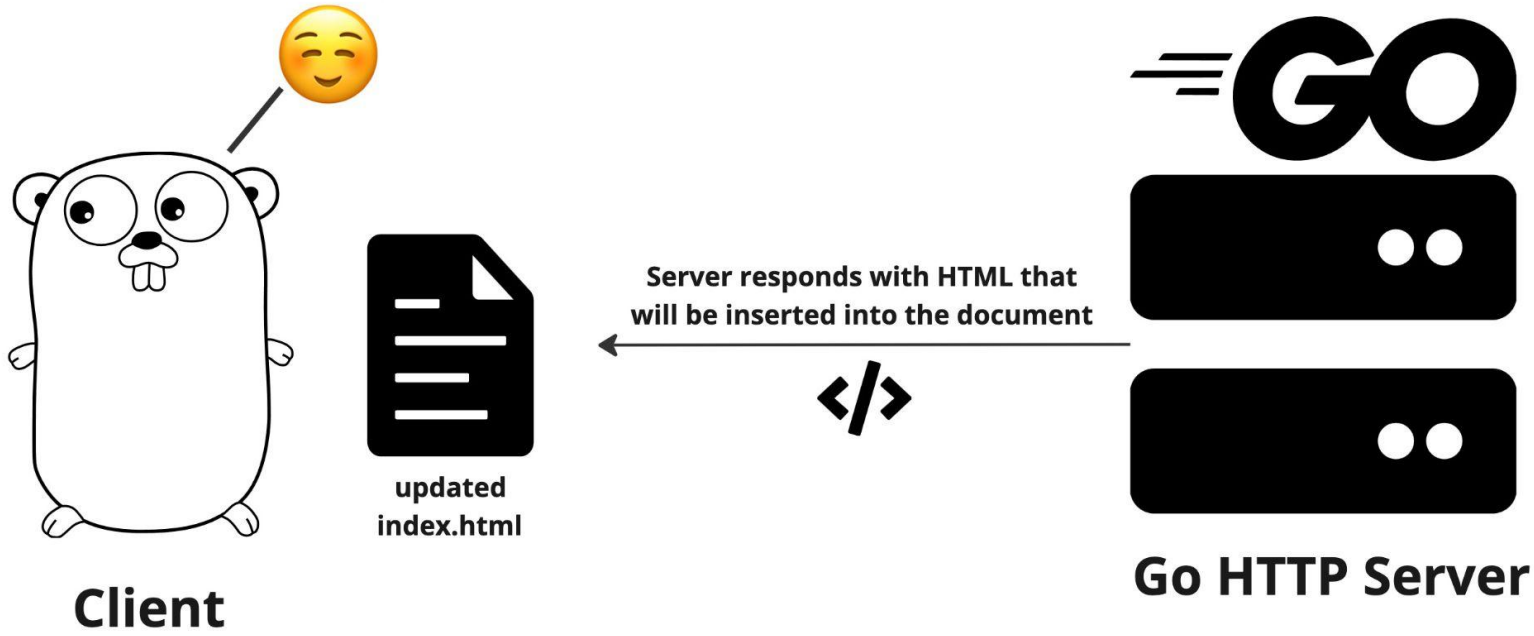
How it works (1 / 3)



How it works (2 / 3)



How it works (3 / 3)



We are going to learn three different technologies including a whole new programming language

- isn't that a bit much?

- developed by Rob Pike (Bell Labs, Plan 9), Ken Thompson (Bell Labs, Unix), Robert Griesemer (V8 Engine) at **Google**, initial release **2012**
- **statically typed**, **AOT-compiled** and **garbage collected** language
- **simple syntax**, easy to learn
- **fast compilation** and **execution times**


```go
func main() {

    file, err := os.Open("test.txt")
    if err != nil {
        fmt.Println("Error:", err)
        return
    }
    defer file.Close()

    fmt.Println("File opened successfully:", file.Name())
}
```

Go(lang): Other Pro Points

- go build tool: Always compiles the program into **a single executable without external dependencies**
- Linter and LSP (gofmt and gopls) easily available
- **Better performance than Node** as a web server [1] [2]

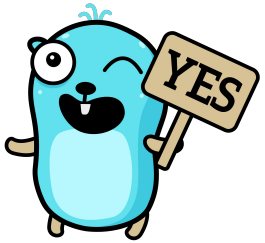
  → Big companies moved at least part of their infrastructure from Node to Go (Uber, Digg and more)

```
~/Projects/Fontys/ESD/esd-2024-goth-stack/Assignment git:(main)±1 (0.993s)
❯ go build .

~/Projects/Fontys/ESD/esd-2024-goth-stack/Assignment git:(main)±2 (0.294s)
❯ du -sh workshop
7.4M    workshop
```

Use cases and weaknesses

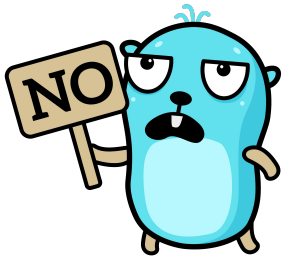
Use cases:



- Has become a popular language for the **cloud** because of its **zero dependencies** and **small memory footprint**
- Big projects like HashiCorp Terraform and **kubernetes** are built in Go

Weaknesses:

- **Embedded**, **Game programming**
- Not a lot of features or syntactic sugar



...Any questions so far?

What is templating?

What is templating?

Templates allow you to **define files with placeholders** in which you can insert data later on, making them **dynamic** and **reusable**:

How does templating look without Templ?

</> TEMPL

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{{ .Title }}</title>
</head>
<body>
    {{ template "header.html" . }}

    <div class="content">
        {{ block "content" . }}{{ end }}
    </div>

    {{ template "footer.html" . }}
</body>
</html>
```

- **No** type-safety ⊘
- **Limited** syntax highlighting ⊘
- **No** inline Go ⊘

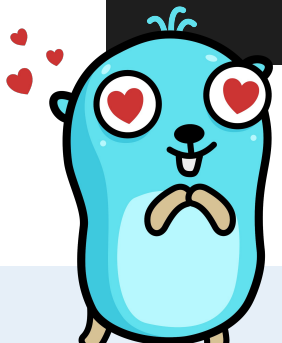
```
{{ define "content" }}
<h2>Welcome, {{ .User.Name }}!</h2>
<p>{{ .User.Message }}</p>
{{ end }}
```

templ

```
package testcall

templ personTemplate(p person) {
    <div>
        <h1>{ p.name }</h1>
        <div style="font-family: &#39;sans-serif&#39;" id="test" data-contents={ `something with "quotes" and a <tag>` }>
            {! email(p.email) }
        </div>
    </div>
}

templ email(s string) {
    <div>email:<a href={ templ.URL("mailto: " + s) }>{ s }</a></div>
}
```



*An **HTML templating language for Go** that has great developer tooling.*

Strengths of Templ

</> TEMPL

- **Strongly typed**: you get compile-time errors instead of runtime errors
- Can **define small, reusable components**
  → This works perfectly with both htmx and TailwindCSS
- Precompiled into go code for **zero runtime overhead**
- Dynamic content is **automatically escaped** by Templ, which increases security

</> htmx

Motivation

- Why should only <a> & <form> be able to make HTTP requests?
- Why should only click & submit events trigger them?
- Why should only GET & POST methods be available?
- Why should you only be able to replace the entire screen?

→ htmx removes these constraints.

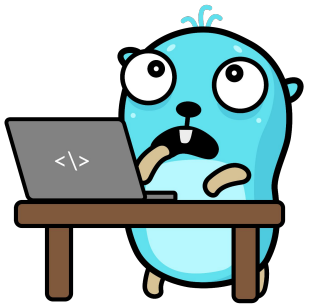
Also, see https://htmx.org/essays/#memes

How does it work in practice?

</> htmx

- Just **load it via a CDN in your html header**, and you're done!
- Use tags such as `hx-get` to issue HTTP requests to the server
- The server responds with HTML
- htmx inserts the result inside of the element that the `hx-target` selector points to (or itself, if there is none)

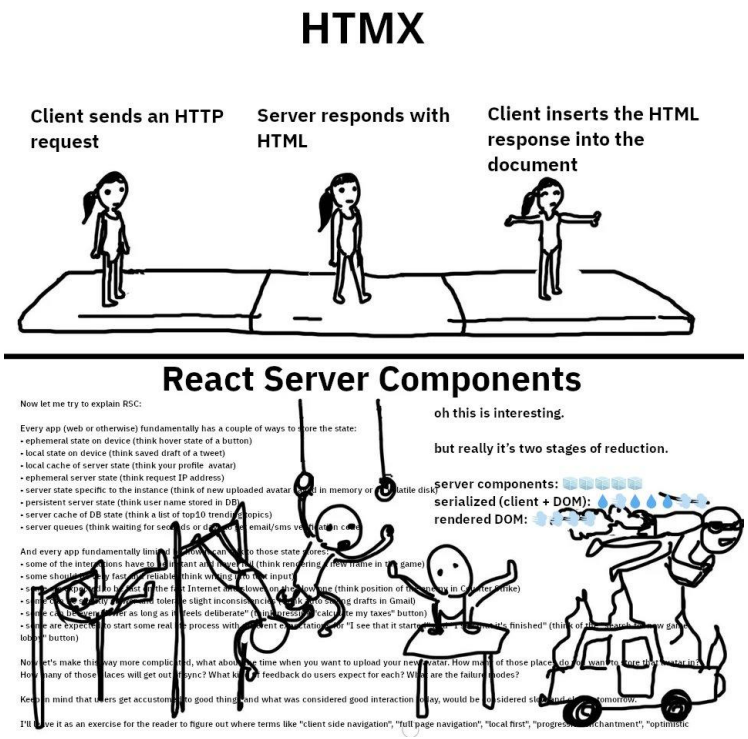
→ Let's see a live demo!



</> htmx

</> htmx

- **Builds upon HATEOAS**
- **Very small** (only ~14KB) and **zero dependencies**
- Has reduced code base sizes by **67%** when compared with react! [1]
- You can use it with react, if you want



When is htmx NOT a good choice?

</> htmx

- **Complex** and highly interactive **client-side state management**, like moving items around via drag and drop, real-time notifications, infinite scrolling
  → possible, but tedious or requires additional JS (probably defeating the purpose)
- **Offline-First Applications**, that only periodically sync with the server for updates
  → not really possible, as htmx requires communication with the server for every operation
- A more serious critique of htmx

(Go)ing further: Extending the GOTH Stack

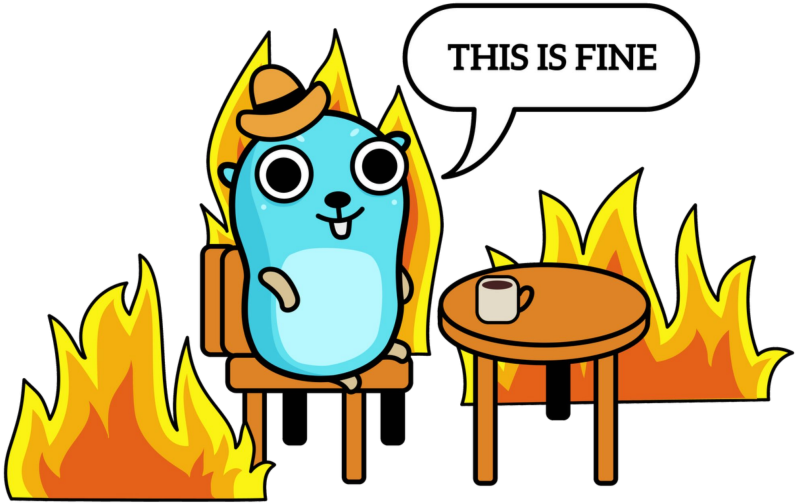
- **TailwindCSS** (We use this!)
- **SQLite**
- **Air** (We use this too!)
- **Echo**
- ...

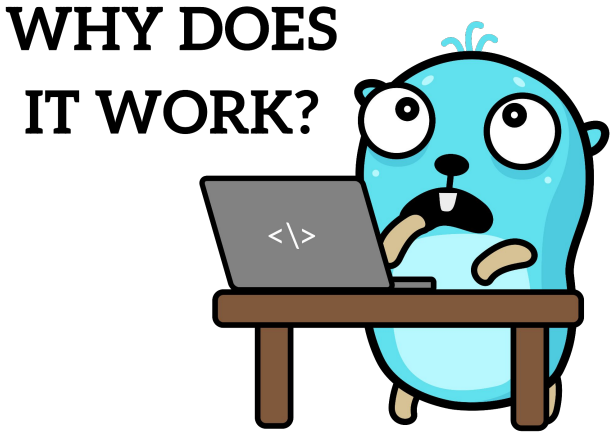
Theory done!
Let’s dive into the practical part!

Let’s combine Go, Templ and htmx to build a small Todo App.

For this part you have time until 12:10 am.

So… how did it go?





Quiz Time!

Did you pay attention?

panic(err)

Now you can prove your knowledge about the GOTH stack and win a small price!

Wrap-Up

Summary

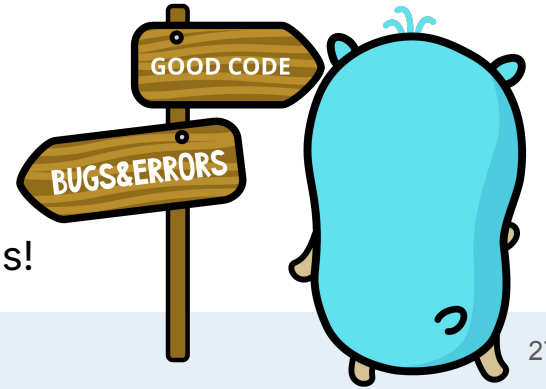
- You learned how to **create simple web apps** with **Go, Templ** and **HTMX**

Where to go from here?

- More information on Go, templ and htmx and resources in the **workshop repository**
  - Includes links to **tutorials**, **forums**, **videos**, **articles** and more.
- Example Project “Cookbook” also in repository. Check it out!

Wanna talk more about the GOTH stack?

- For questions, discussions and memes, just ask us after class!



Thank you for your participation!

We hope you had fun, learned something, and would consider the GOTH stack for your future projects!

We're looking forward to your (positive) feedback!

Special thanks to MariaLette for the free to use gopher icons
https://github.com/MariaLetta/free-gophers-pack/tree/master